Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department

Deep Learning/Odd Sem 2023-23/Experiment 3b

Name : Rupesh Dhirwani	Class/Roll No. : D16AD - 10	Grade :

Title of Experiment : Autoencoder for image denoising

Objective of Experiment:

To evaluate the effectiveness of Autoencoders (AE) for image denoising Autoencoders are a type of artificial neural network commonly used for data compression and feature learning

Outcome of Experiment: Thus we used an Autoencoder For image Denoising

Problem Statement: Images acquired from various sources often suffer from noise, which can degrade their quality and impact their utility for tasks such as image recognition and analysis Traditional image-denoising techniques may not always provide satisfactory results. This experiment aims to address the following questions:

- Can autoencoders effectively denoise images by learning and capturing essential features?
- How does the performance of autoencoders compare to traditional image denoising methods in terms of noise reduction and preservation of image quality?

Description/Theory:

Autoencoders, a type of neural network, are commonly used in tasks like image denoising. They work by training a network to reduce noise in noisy images and restore them to cleaner versions.

Autoencoder Basics: Autoencoders consist of an encoder and a decoder. The encoder compresses input data into a lower-dimensional representation, while the decoder reconstructs the original input from this representation

Training Process:

During training, noisy images and clean versions are used. The network learns to minimize the difference between its reconstructions and clean images.

Denoising:

After training, the autoencoder can denoise new noisy images by passing them through the network.

Optimization:

Tuning hyperparameters, like network architecture and learning rate, is crucial for optimal performance.

Variations:

Variations like convolutional autoencoders (CAES) or denoising autoencoders (DAES) enhance denoising capabilities by considering spatial information or explicitly removing noise

PROGRAM:

```
import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow.keras import layers

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Model

def preprocess(array):

#     Normalizing the supplied array and reshapes it into the appreciate format.

    array = array.astype("float32") / 255.0

    array = np.reshape(array, (len(array),28, 28, 1))

    return array

def noise(array):

#     Adds random noise ro each image of th supplied array

    noise_factor = 0.4

    noisy_array = array + noise_factor * np.random.normal(loc = 0.0, scale = 1.0, size =
array.shape)

    return np.clip(noisy_array, 0.0, 1.0)

def display(array1, array2):

#     Displays ten random images from each one of the supplied arrays.

    n = 10
```

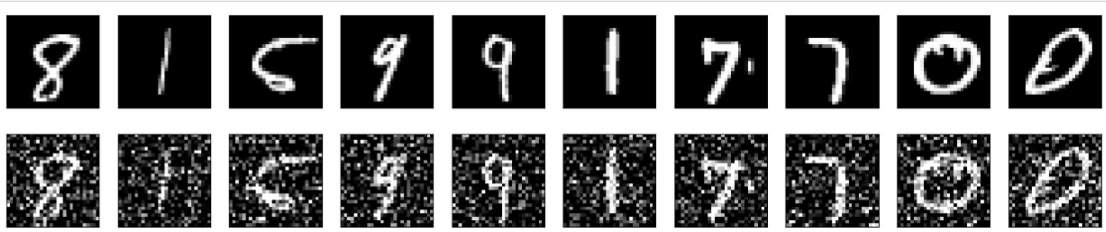
```
    indices = np.random.randint(len(array1), size = n)
    images1 = array1[indices, :]
    images2 = array2[indices, :]
    plt.figure(figsize=(20,4))
    for i, (image1, image2) in enumerate(zip(images1, images2)):
        ax = plt.subplot(2,n,i+1)
        plt.imshow(image1.reshape(28,28))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

        ax = plt.subplot(2,n,i+1+n)
        plt.imshow(image2.reshape(28,28))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
plt.show()
(train_data, _), (test_data, _) =  mnist.load_data()
train_data = preprocess(train_data)
test_data = preprocess(test_data)
noisy_train_data = noise(train_data)
noisy_test_data = noise(test_data)
```

```
display(train_data, noisy_train_data)

input = layers.Input(shape = (28, 28, 1))

x = layers.Conv2D(32, (3, 3), activation = "relu", padding = "same")(input)

x = layers.MaxPooling2D((2, 2), padding = "same")(x)

x = layers.Conv2D(32, (3,3), activation = "relu", padding = "same")(x)

x = layers.MaxPooling2D((2,2), padding = "same")(x)

x = layers.Conv2DTranspose(32, (3, 3), strides = 2, activation = "relu", padding = "same")(x)

x = layers.Conv2DTranspose(32, (3, 3), strides = 2, activation = "relu", padding = "same")(x)

x = layers.Conv2D(1, (3,3), activation = "sigmoid", padding = "same")(x)

autoencoder = Model(input, x)

autoencoder.compile(optimizer = "adam", loss = "binary_crossentropy")

autoencoder.summary()

autoencoder.fit(

    x = train_data,

    y = train_data,

    epochs = 1,

    batch_size = 128,

    shuffle = True,

    validation_data = (test_data, test_data),

)

predictions = autoencoder.predict(test_data)

display(test_data, predictions)
```

```
autoencoder.fit(
    x = noisy_train_data,
    y = train_data,
    epochs = 1,
    batch_size = 128,
    shuffle = True,
    validation_data = (noisy_test_data, test_data),
)
predictions = autoencoder.predict(noisy_test_data)
display(noisy_test_data, predictions)
```

OUTPUT:

```
In [10]: display(train_data, noisy_train_data)
         input = layers.Input(shape = (28, 28, 1))
```



```
In [17]: display(test_data, predictions)
```

```
In [20]: display(noisy_test_data, predictions)
```

Results and Discussions:

The provided code demonstrates the use of a denoising autoencoder on the MNIST dataset to remove noise from images. After training, the autoencoder successfully denoises the test images.

Discussion:

The autoencoder architecture successfully denoises images by training on noisy data. This technique is applicable in broader contexts, including medical imaging and photo enhancement. Customization of architecture and training parameters can adapt the code for specific tasks